Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali (L-Z)

Appello del 18/4/2008

Esercizio 1 (4 punti)

Dichiarazione ed utilizzo di metodi e variabili membro in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M, int N) {
  int i=M;
  while(i<N) {
    if(V[i]>V[0]) return V[i];
    i++;
  }
  return V[0];
}

int g(int V[], int M, int N) {
  int j=0, res=0;
  while(j<N)
    res+=f(V, j++, M);
  return res;
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `M<N` e si consideri il caso peggiore).
2. Calcolare la complessità in passi base della funzione `g` nei termini dei parametri `M` ed `N`.
3. Calcolare la complessità asintotica della funzione `g` nei termini dei parametri `M` ed `N`.

Esercizio 3 (5 punti)

Per fronteggiare un'eventuale invasione, l'agenzia governativa DLB del pianeta Croma sta raccogliendo i dati relativi agli avvistamenti degli alieni del pianeta Eldenia. Per questo, ogni avvistamento viene caratterizzato dal luogo e dalla data in cui è avvenuto.
Si scriva una classe `Avvistamento` per l'agenzia governativa DLB che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi per accedere alle proprietà dell'avvistamento.
3. Presenti il metodo `toString` che fornisca una descrizione testuale dell'avvistamento.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Avvistamento` (l'uguaglianza va verificata sia sul luogo che sulla data).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Avvistamento`, dando la precedenza all'avvistamento più recente.

Esercizio 4 (8 punti)

Si scriva una classe `Alieno` che memorizzi le informazioni relative agli avvistamenti di ogni razza del pianeta Eldenia. Per ogni razza aliena occorre memorizzare, oltre ad una lista che contenga i dati sugli avvistamenti, il nome della razza ed una stringa che ne contiene la sequenza DNA. La classe `Alieno` deve inoltre:

1. Possedere un opportuno costruttore (inizialmente nessuna razza può essere avvistata prima della sua creazione).
2. Presenti metodi per accedere al nome ad alla sequenza DNA della razza aliena.
3. Possedere il metodo `nuovoAvv` che, dato un oggetto `Avvistamento`, lo inserisca nella lista, mantenendo tale lista ordinata secondo il punto 5. dell'Esercizio 3.
4. Possedere il metodo `ultimoAvv` che, dato un luogo, restituisca l'avvistamento più recente avvenuto in tale luogo.
5. Presentare il metodo `quanti` che restituisca il numero di avvistamenti della razza aliena.
6. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Alieno` (l'uguaglianza va verificata unicamente sulla sequenza DNA).
7. Presentare il metodo `toString` che restituisca una descrizione della razza aliena.

Esercizio 5 (7 punti)

Si scriva un'applicazione per l'agenzia governativa DLB che:

1. Crei un insieme di oggetti di tipo `Alieno`.
2. Lette da tastiera le informazioni relative ad un nuovo alieno, provveda ad inserire tale oggetto all'interno dell'insieme (a meno che all'interno dell'insieme non esista già un oggetto uguale).
3. Lette da tastiera le informazioni relative ad un nuovo avvistamento, provveda ad aggiungere tale `Avvistamento` all'interno della lista dell'`Alieno` di cui al punto 2.
4. Letta da tastiera la descrizione di un luogo, stampi su video le informazioni relative all'`Alieno` avvistato più di recente in tale luogo.
5. Stampi su video le informazioni relative all'`Alieno` avvistato più di frequente.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `boolean leggiBoolean()` Legge un boolean (delimitato da spazi).
- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	1
`while`	$N - M + 1$
`if(V[i]>V[0])`	$N - M$
`i++`	$N - M$
Totale	$3N - 3M + 2$

Domanda 2:

2 assegnamenti	2
`while`	$N + 1$
chiamata di `f`	N
complessità di `f`	$3MN - 3N^2/2 + 7N/2$
Totale	$3MN - 3N^2/2 + 11N/2 + 3$

Domanda 3:
Complessità asintotica: $O(MN)$

Soluzione Esercizio 3

```
class Avvistamento implements Comparable<Avvistamento> {
  private String luogo;
  private int g, m, a;

  public Avvistamento(String luogo, int g, int m, int a) {
    this.luogo=luogo;
    this.g=g; this.m=m; this.a=a;
  }

  public String getLuogo() { return luogo; }
  public String getData() { return ""+g+"/"+m+"/"+a; }
  public String toString() { return luogo+" "+getData(); }

  public boolean equals(Object o) { return equals((Avvistamento) o); }
  public boolean equals(Avvistamento a) {
    return luogo.equals(a.luogo)&&getData().equals(a.getData()); }

  public int compareTo(Avvistamento v) {
    int ret=(v.a-a);
    if(ret==0) ret=(v.m-m);
    if(ret==0) ret=(v.g-g);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Alieno {
  private List<Avvistamento> avv;
  private String nome, DNA;

  public Alieno(String nome, String DNA) {
    this.nome=nome;
    this.DNA=DNA;
    this.avv=new LinkedList<Avvistamento>();
  }

  public String getNome() { return nome; }
  public String getDNA() { return DNA; }

  public void nuovoAvv(Avvistamento a) {
    int i=0;
    while((i<avv.size())&&(avv.get(i).compareTo(a)<0)) i++;
    avv.add(i, a);
  }

  public Avvistamento ultimoAvv(String luogo) {
    for(Avvistamento a:avv)
      if(a.getLuogo().equals(luogo)) return a;
    return null;
  }

  public int quanti() { return avv.size(); }
  public boolean equals(Object o) { return equals((Alieno) o); }
  public boolean equals(Alieno a) {
    return DNA.equals(a.DNA); }

  public String toString() { return nome+" ("+DNA+")\n"+avv; }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Alieno> s=new TreeSet<Alieno>(); // domanda 1
    Alieno r=new Alieno(Lettore.in.leggiLinea(),
      Lettore.in.leggiString());

    s.add(r); // domanda 2
    r.nuovoAvv(new Avvistamento(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt())); // domanda 3
    String luogo=Lettore.in.leggiLinea();
    Alieno ultimo=null;
    for(Alieno a:s) {
      Avvistamento avv=a.ultimoAvv(luogo);
      if(avv!=null)
        if((ultimo==null)||(ultimo.ultimoAvv(luogo).compareTo(avv)>0))
          ultimo=a;
    }
    if(ultimo!=null) System.out.println(ultimo); // domanda 4
    Alieno freq=null;
    for(Alieno a:s)
      if((a.quanti()>0)&&((freq==null)||(freq.quanti()<a.quanti())))
        freq=a;
    if(freq!=null) System.out.println(freq); // domanda 5
  }
}
```